GEORGIA INSTITUTE OF TECHNOLOGY

School of Electrical and Computer Engineering

ECE-6276-A DSP Hardware System Design (Fall 2018)

Lab 7: Decimated-in-frequency FFT using the Butterfly Technique

---

Assigned Date : Wednesday, October 10, 2018

Due Date : 23:55, Friday, October 19, 2018

---

1 Introduction

FFT is a method of performing Discrete Fourier Transform which basically represents a time domain signal in terms of its components at different discretized frequency levels (frequency bins). In this lab, we will design a Decimated-in-frequency (DIF) FFT Transform system for `n=8` (See Fig.1). The twiddle factors are 9 bits wide and will be provided. Also, note that the multiplication and addition are signed operations. The template for FFT is provided in `fft_top.vhd` in `./src` directory. Also, note that some useful data structures have been provided in `fft_pkg.vhd` in `./src` directory. Feel free to add needed data structures into `fft_pkg.vhd` on your own. `http://vhdlguru.blogspot.com/2011/06/non-synthesisable-vhdl-code-for-8-point.html` is a good reference source for the design. Although the code given in the link is **non-synthesizable**, you can base your final code on this.

The explanations of the input and output ports are given below.

- `clk` => Clock signal. We use rising edge for flops.
- `rst` => Reset signal. Note that the sequential elements that are used in your design should have **synchronous** active high reset. i.e. if the reset is pulled high, the flops should be reset at next rising edge of clk.
- `in_valid` => `in_valid` is the qualifier for the data. When `in_valid = 1`, the data at the same cycle are valid.
- `next_in` => `next_in` is generated by the design and tells the testbench that, if `next_in = 1`, the testbench can feed next data set, otherwise the testbench has to hold the current valid data.

- `data_in` => The input data port from which data are passed to the design. Note that the real 8-bit fix-point data are passed to the design serially.
- `data_real_out and data_imag_out` => Outputs from the design. After the complex operations, the real data which are passed into the input are transformed into real and imaginary components.
- `out_valid => valid signal` for the output. When `out_valid = 1`, the outputs `data_real_out` and `data_imag_out` are valid. This information is useful to the test-bench which captures the data only when output is valid. The type of `data_real_out` and `data_imag_out` is defined in `fft_pkg.vhd`. They are arrays of 8 elements with each element being 8 bit wide. The real and imaginary outputs in the reference output file are ordered in the natural ascending order: $X[0], X[1], \ldots, X[7]$.

Note that there is no `next_out` signal for the design.

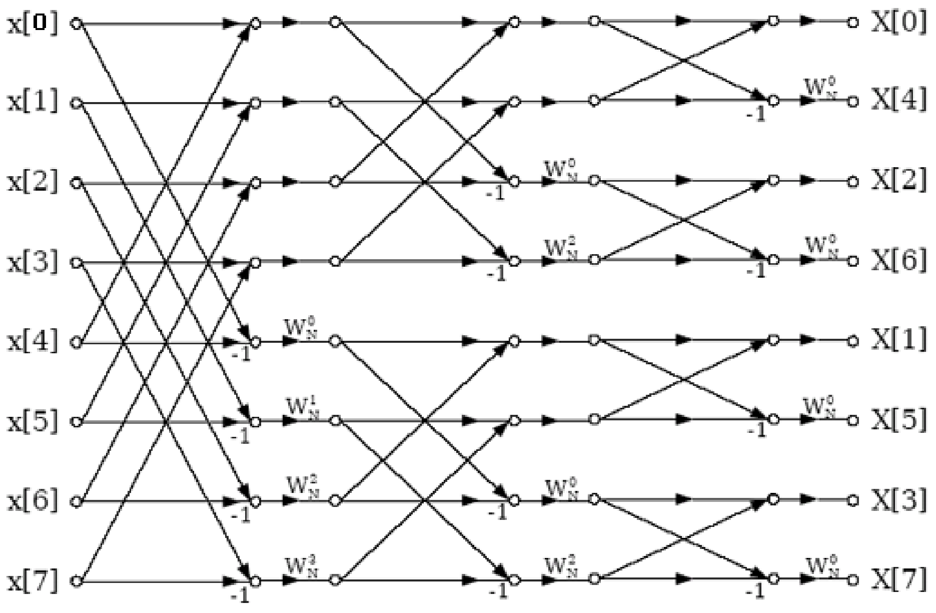

Figure 1 DIF 8 point FFT (Source: `https://www.quora.com/What-is-the-difference-between-decimation-in-time-and-decimation-in-frequency`)

Table 1 Twiddle Factor Table

	Real	Imaginary
W_8^0	011111111	000000000
W_8^1	010110100	101001100
W_8^2	000000000	100000001
W_8^3	101001100	101001100

Table 1 lists the twiddle factors which should be used in the design. The other twiddle factors are not listed as they will not be used in the calculation of FFT.

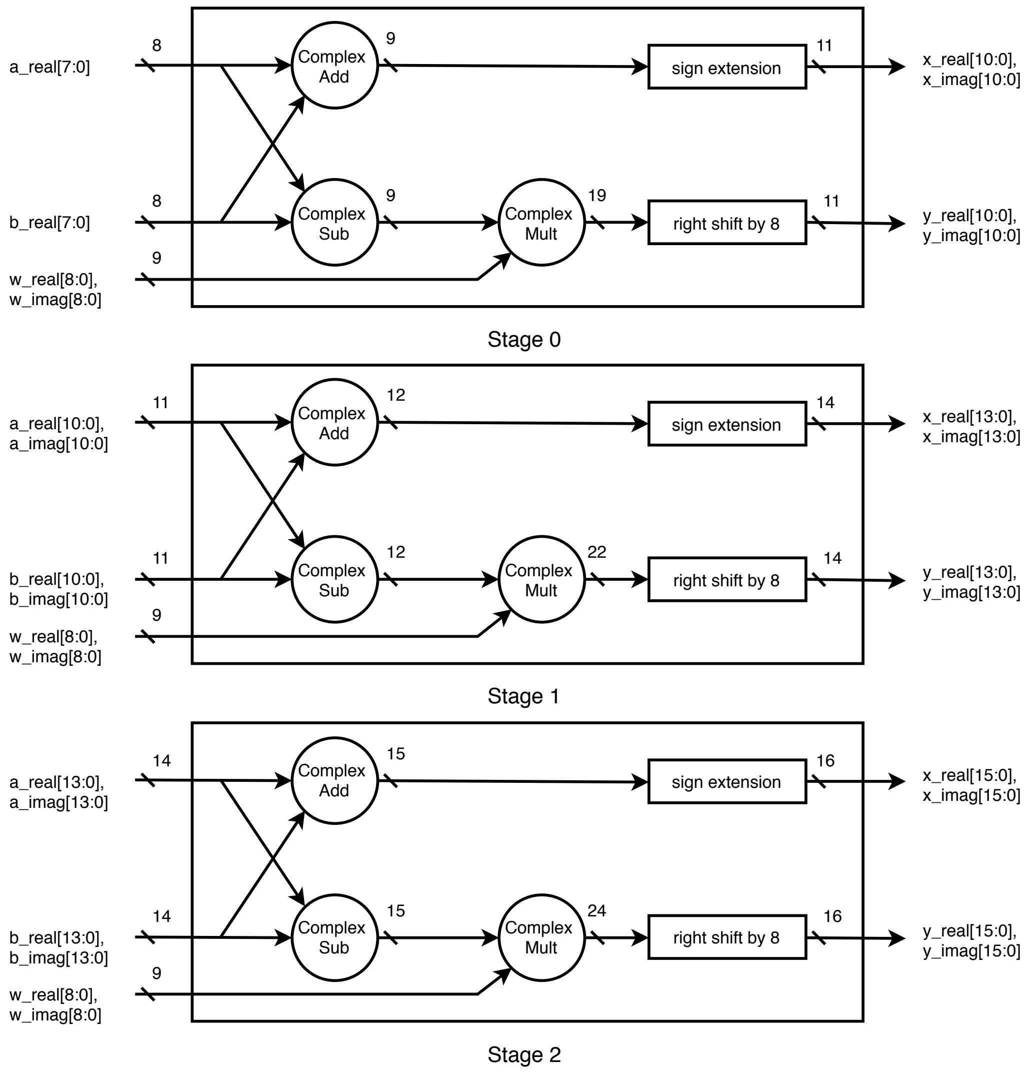

Figure 2 DIF 8 point FFT

Taking care of the bit-growth and doing the correct wire mapping will be among the biggest challenges in this lab. The block diagram Fig. 2 helps you keep track of the bit-growth. Note that there is no imaginary part at the input of the first stage.

Useful Hints: The data types declared in the `fft_pkg.vhd` will be helpful in creating intermediate signals. Please note that the very first file that you compile should be `fft_pkg.vhd` because the `tb_fft_top.vhd` uses some datatypes declared in the package.

This system can be implemented in numerous ways with different latencies,area and power. You can use any architecture as long as you can clearly draw the block diagram of your architecture in the report. Fig. 1 shows a straight-forward pipelined architecture. If you are using this, you don't need to draw the block diagram. However, the design may be very bad for area but is quick. So the testbench gives a data sample every cycle, if possible. In your design, you can choose to use a memory/buffer which stores all these samples and will parse it to main logic at a speed which is decided by the latency of your architecture. This additional memory overhead will not be considered for grading deduction.

There is a sample input file from which the data is read into the testbench. The testbench dumps out an output file. Compare this output with the reference output file provided with the lab. The output will be the same for any architecture. The grading will take into consideration the functional correctness, utilization (area), and timing of the design.

For synthesis and implementation, since our design has too many I/O ports, it will run out of I/Os if we map it on Basys 3. Therefore, please select the chip `xcku115-flva2104-3-e` for synthesis which has a large amount of resources and the chance of running out of resources is very small.

Three re-submissions are allowed for this lab.

2 Instructions

1. Draw the block diagram of the architecutre you will be using. (If you are using the same architecture as the figure above, then you don't need to draw it)
2. Complete your design in VHDL. You can use multiple design files this time, but the top file should be `fft_top.vhd`.
3. Run the simulation for `fft_top.vhd` using the provided testbench `tb_fft_top.vhd` to match your output file `output.txt` with the reference `output_ref.txt`.
4. Run the synthesis and implementation of your design `fft_top.vhd` in Vivado using the provided constraint file `constraints.xdc`.

3 Deliverables

1. Create a PDF report containing:
   - Block diagram of your design (If you are using the same architecture as the figure above, then you don't need to draw it). If your design has a very small area, 5 extra points will be given for this bullet. But the total score will not exceed 100.

- Tables of
  (a) Resources (Only "Slice Logic", "IO and GT Specific", "Primitives") from `fft_top_utilization_placed.rpt`
  (b) Power (Dynamic and Static) from `fft_top_power_routed.rpt`
  (c) Worst Negative Slack, Worst Hold Slack ("Design Timing Summary") from `fft_top_timing_summary_routed.rpt`
- (15% of grade) Answer the question: What is the rationale behind choosing the values of the twiddle factor as given in the table? Note that the twiddle factors are signed numbers of resolution 1.8.

  Hint: Think of how you do fractional multiplication for fixed point numbers. We do a right shift of 8 after each stage of the butterfly. Analyze by showing your calculation for twiddle factors.

The name of the PDF should be of the form `lab7_firstname_lastname.pdf` , e.g. `lab7_george_burdell.pdf` .

2. (75% of grade for this bullet and the next) The completed design file `fft_top.vhd` . **(No latch generated; synchronous reset, sensitivity signal list correct) Note that if the design is not verified, the block diagram and implementation will not be considered**
3. Other modules (if applicable)
4. The simulated output file `output.txt` in the "run" folder.
5. The simulated output cycle file `output_cycle.txt` in the "run" folder.
6. (10% of grade) The implemented area (utilization), power and timing reports, namely,

   `fft_top_utilization_placed.rpt`

   `fft_top_power_routed.rpt`

   `fft_top_timing_summary_routed.rpt`

Move all of these files into a folder called `lab7_firstname_lastname_gtID` . Zip the folder and upload the archive `lab7_firstname_lastname_gtID.zip` on T-Square (eg. `lab7_george_burdell_123456789.zip`). The first name and last name should be all in lower case. **Please strictly follow this naming convention. Otherwise my script will not work and you might get points deduction.**

Note: Late submissions are not accepted. In case of extraordinary circumstances, written permission must be obtained from Dr. Madisetti.